个性化视频推荐系统的设计与实现

摘　要

各种视频网站以及小视频的出现，自然的会产生大量的视频数据，产生的问题的是：用户怎么从这么多的视频数据里面选择自己喜欢的视频？我们用推荐系统来解决此问题。

本论文研究的是个性化视频推荐系统，主要是收集用户的爱好，以及 web 的操作日志，例如用户的观看记录，观看后对视频的评分等信息。然后通过基于用户的协同过滤算法给用户推荐出符合用户的视频。

系统分为以下几个模块，用户模块:用户的相关操作，日志模块:收集用户在界面的操作日志，推荐模块：按照协同过滤算法给用户推荐视频，管理模块：对视频和用户的管理，例如添加新视频，修改视频相关操作，定时任务模块：计算推荐数据和执行相关的定时任务。各个模块通过共享数据库来衔接。系统前台使用 Bootstrap, jQuery, 后台使用 Python 语言，Django web 框架，采用 Oracle 数据库来开发。

关键词:　视频　　协同过滤　　模块　　Django　　推荐系统

The design and implementation of personalized video recommendation system

ABSTRACT

The emergence of various video websites and small videos naturally generates a large amount of video data. The question arises: How do users choose their favorite videos from so many video data? We use a recommendation system to solve this problem.

This dissertation studies the personalized video recommendation system, which mainly collects the user's hobby and the web's operation log, such as the user's viewing record, the rating of the video after watching, and other

information. Then the user is recommended to match the user's video through a user-based collaborative filtering algorithm.

The system is divided into the following modules: user module: user-related operations, log module: collecting user's operation logs in the interface, recommendation module: recommending videos to users according to certain algorithm, management module: managing videos and users, for example add new video, modify video-related operations, timing task module: calculate recommended data and perform related timing tasks. Each module connects through a shared database. The system front-end uses Bootstrap, jQuery, back-end Python language, Django web framework, and development using Oracle database.

Key Words: Video Collaborative filtering Module Django Recommendation system.

目　　录

第一章　绪论

1.1 研究背景和意义

如今我们生活在一个大数据的时代，智能时代。网络的应用使得资源的产生和分享很容易。互联网用户不断的在增加，这些用户每天会产生大量的博客、视频、图片以及与个人兴趣爱好相关的数据。我们从原来的信息缺乏进入到信息过载[1]时代。对于产生这些信息的人来说，怎么让自己的信息更快的传播，对于消费这些信息的人，怎么找到符合自己的信息都是极大的挑战。

在人们面对海量的数据无法筛选出自己有用的信息时，出现了搜索引擎[2]，从以前的文字搜索转变为如今的语音搜索，这都给用户带来了很多的便利。但是，搜索引擎是建立在用户的需求已知情况下，并且搜索结果很多是相近的，需要用户通过自己的主观意识去判断，那个是自己需要的。如果用户的需求不明确，就是为了放松或娱乐，主要得符合自己的兴趣爱好，那么有什么好的办法吗？

推荐引擎[3]就可以弥补搜索查找的这种不足。它知道你的爱好，知道你喜欢什么，因此可以快速的推荐出与你兴趣爱好对应的信息，方便了用户的筛选，减少用户看到重复的信息和厌烦的广告甚至不健康的信息，增加了体验效果。

推荐引擎还可以处理的一个问题是信息过多。我们都生活在互联网时代，网络的应用与生活息息相关。去超市不用带现金，坐公交可以不带公交卡，吃饭可以不去饭店等等，这么多的应用因此会产生大量的数据。推荐引擎能从这么多的数据中筛选自己有用的信息。

过去的视频网站例如优酷，腾讯视频，爱奇艺，搜狐等，主要是根据视频的分类和搜索引擎来搭建，随着推荐系统的发展，逐渐的在原来的基础之上都添加了推荐功能。还有很热的小视频抖音，快手，火山等，用户随手一拍就可以产生一个。它们主要以推荐为主，抓住了用户的兴趣。

每个人所属的行业不同，因此关注的领域不一样，不论推荐系统是推荐什么，主要的问题是把用户关注的领域各方面都能推荐给用户，如果是以领域划分的推荐系统，那么应该把这一领域的历史，相关的资料文章，以及涉及到人物等等都可以推荐给用户。这样对信息的传播也是很有价值，当然这些在技术实现上也需要很大的研究和发展。如果我们是推荐物品，那么抓住用户的兴趣是非常重要，达到个性化的推荐，有些系统的推荐是针对所有的用户，与用户的兴趣爱好并不相符，进而不是个性化的。应该站在用户的角度多考虑问题，减少用户的输入，提高自动化推荐的功能。最后还要能建立在用户以前的会话基础上，这样才能达到个性化的推荐。

1.2 国内外发展状况

1.2.1 推荐系统国外研究现状

“个性化推荐系统”一词是在 1995 年 3 月[4]，斯坦福大学的 MarkoBalabanovic 等人在美国人工智能协会上所提出的。从此开启了推荐系统在互联网方面的大门。学术界在这方面也投入了大量的资源，取得了不错的成果，尤其对推荐算法的研究。目前为止，数据库、数据挖掘、机器学习方面的重要国际会议都有大量与推荐系统相关的研究成果发表。

美国的 Amazon[5]，不管是在过内还是国外，在推荐系统方面是起步比较早的公司，并且也做的比较成熟。在 1998 年运用基于物品的协同过滤算法搭建了推荐系统。由于电子商务推荐对实时性要求较高，Amazon 采用了一种称为商品到商品的协同过滤算法，此算法可以在海量的数据中实时计算出推荐数据。

2003 年在 IEEE Internet Computing 上发表了基于物品的协同过滤算法[6]，该算法逐渐被传开。YouTube 的人员将他们的用户归为三类，第一，用户只在网站上观看一些与某一特定话题相关的视频；第二，在别处看到某一视频的部分，去网站看完整视频；第三，没有明确目的，只是看可以使自己喜悦的视频。为了满足第三种类型的人，2010 年 YouTube 也使用协同过滤算法做他们的视频推荐[7]。

Netflix 也有他们的推荐系统，并且 Netflix 的人员声称他们的观看记录大多是来自他们的推荐系统推荐的电影[8]，推荐系统给用户带来了极大的便利。

1.2.2 推荐系统国内研究现状

国内的互联网虽然没有国外的起步早，但是近几年发展非常迅速。推荐系统方面也不例外。比如在国内非常受关注的个性化推荐网站豆瓣[9]。在 2009 年 7 月，我国出现了在推荐系统方面的研究团队。2011 年 9 月，李彦宏将推荐引擎等划入未来发展的方向[10]。渐渐的国内在推荐系统应用方面也逐步扩大，表 1.1 展示了国内部分在推荐系统方面的应用。

表 1.1 国内推荐系统应用

领域	推荐系统
电子商务	淘宝网、京东商城、当当
新闻	今日头条
视频	爱奇艺、优酷、豆瓣、抖音
阅读	知乎、简书
音乐	网易云音乐、QQ 音乐
其他	社交软件推荐可能认识的人

在中国知网上，通过关键词“个性化推荐系统”去检索，会出现很多的相关文献，图 1.1 展示了 2001 年到 2018 年学术界在这方面发表的文献的数量，可以看出基本呈指数的形式在上升。体现了推荐系统应用的广泛性以及对推荐系统的关注度。

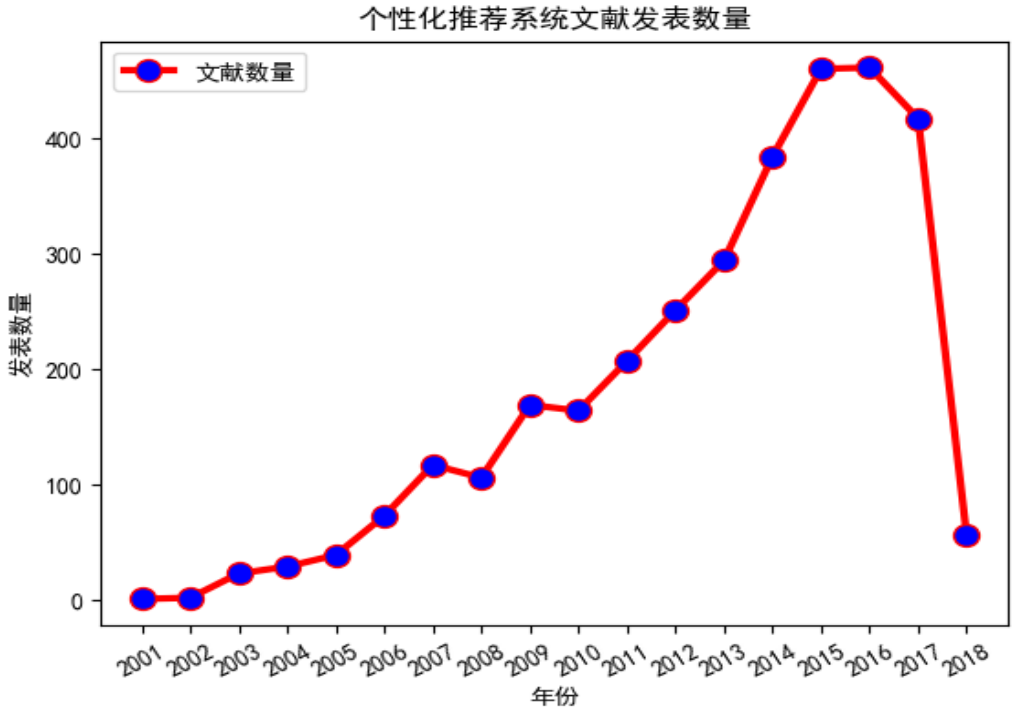


图 1.1 个性化推荐系统学术关注度

在国内，在推荐系统方面的研究主要集中在推荐算法和模型的构建[11]，现在我们已经进入了大数据的时代，互联网上可以爬去大量的数据，通过机器学习然后训练构建切实际的模型。算法的研究主要是改进算法，例如范波[12]等人应用了一种基于用户间多相似度的协同过滤算法来改进单一的评分相似度的缺点。

1.3 论文内容结构

此论文的讲解由五部分构成，各部分的基本内容如下：

第一章，讲解研究推荐系统的原因，研究推荐系统有什么意义，以及国内外在推荐系统方面的研究状况进行概述。

第二章，讲解个性化视频推荐系统开发涉及到的技术，以及推荐系统普遍的框架构成，和推荐系统中常用的推荐算法。

第三章，从软件工程的角度考虑问题，讲解系统的需求分析，设计，数据库的设计。

第四章，从开发实现的角度讲解，给出具体的实现流程，落实于实现思路。

第五章，为了系统能够稳定安全的运行，进行系统测试。

1.4 本章小结

本章主要讲解研究个性化视频推荐系统有什么意义，为什么要研究，以及推荐系统在国内外的发展状况，第三小节对整个论文的结构内容进行了概括，以便读者能够提前了解论文的讲解思路。

第二章　个性化视频推荐系统相关技术

2.1 推荐系统开发环境及技术

个性化视频推荐系统用面向对象的解释性 Python 语言开发[13]，在强大的 PyCharm 集成开发环境软件的支持下，利用开放源代码的 Django Web 框架[14]所开发，这样能够降低设计的成本，节省开发时间。系统开发和运行环境介绍如下：

硬件环境：CPU 类型为 i5, 内存为 8G

数据库系统：oracle 11.2.0

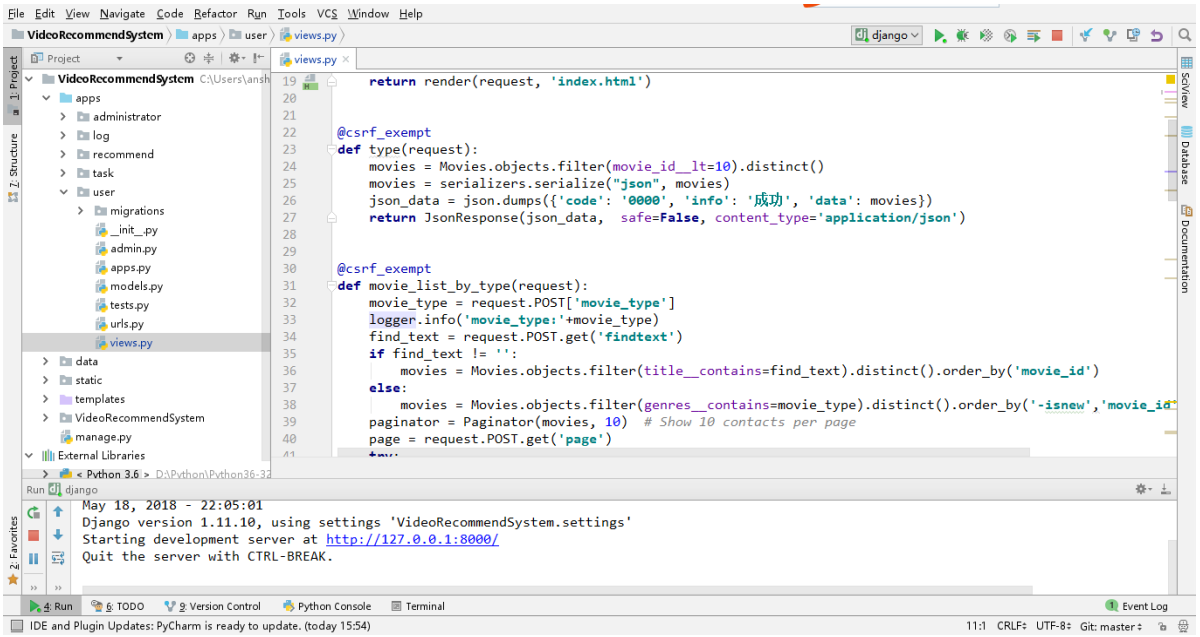
开发工具：HTML、JavaScript、CSS、Python3.6、PyCharm

项目框架：前台 BootStrap、后台使用 Django Web 框架

代码管理：Github

2.1.1 PyCharm 简介

PyCharm 是 JetBrains 公司开发的一款商业的 Python 集成开发环境，如图 2.1, 编写代码会给出提示，不同的语法颜色不同，例如关键字，字符串等字体颜色不一样，并且集成版本控制管理工具，对开发人员的操作提供便利，最主要的是有调式功能，对每一个开发人员，调式是开发人员必备技能，很快能定位到错误的位置或者能深入的了解程序运行的过程。我们的系统是基于 web 的，该编辑器还支持 web 开发。




```
    return render(request, 'index.html')


@csrf_exempt
def type(request):
    movies = Movies.objects.filter(movie_id__lt=10).distinct()
    movies = serializers.serialize("json", movies)
    json_data = json.dumps({'code': '0000', 'info': '成功', 'data': movies})
    return JsonResponse(json_data,  safe=False, content_type='application/json')


@csrf_exempt
def movie_list_by_type(request):
    movie_type = request.POST['movie_type']
    logger.info('movie_type:'+movie_type)
    find_text = request.POST.get('findtext')
    if find_text != '':
        movies = Movies.objects.filter(title__contains=find_text).distinct().order_by('movie_id')
    else:
        movies = Movies.objects.filter(genres__contains=movie_type).distinct().order_by('-isnew','movie_id
    paginator = Paginator(movies, 10)  # Show 10 contacts per page
    page = request.POST.get('page')
```

图 2.1 PyCharm 工作平台

2.1.2 Django 简介

基于 Python 的 web 框架有 Django，Flask，Tornado，Bottle 等等,本系统采用 Django 框架，是一个开放源代码的 Web 型框架。使用了 MTV 的架构模式，即模型 M（model），模板 T（template）和视图 V（view）。提供了很多的模块，最主要的有 admin 模块，可以很好的管理后台，并且能方便的集成第三方的插件。还有它内置 web 服务器，这样给开发测试带来了极大好处。如图 2.2 是 Django 的工作模型。

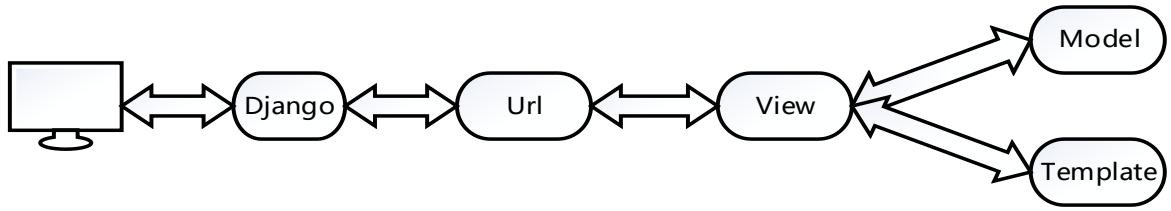


图 2.2 Django 工作模型

2.2 推荐系统架构

推荐系统一般分为三个模块，推荐模块，UI 模块，日志模块。推荐引擎是推荐系统的大脑，这个需要根据实际的系统不断查漏补缺，优化推荐算法。大型的推荐系统都不会用单一的推荐算法，因为这样推荐的效果非常差。在数据处理方面也会用离线计算和实时计算相结合的方式，最后混合推荐的数据，这个可以在后台通过权值混合或者在前台交给用户来切换。UI 部分由三部分构成：第一，展示推荐结果，视频资源的话可以是缩略图标题，新闻的话可以是标题关键词等等，视具体情况而定。第二是为什么推荐此项目，例如给出评分或关键词，可以增加用户的信任度。第三是回馈，让用户与系统交互，动态更新推荐结果。日志模块主要采集用户反馈的结果，为推荐模块提供有效的数据，方便训练符合用户特点的模型给用户推荐恰当的物品。如图 2.3 是推荐系统一般架构，包含各个模块之间的联系。

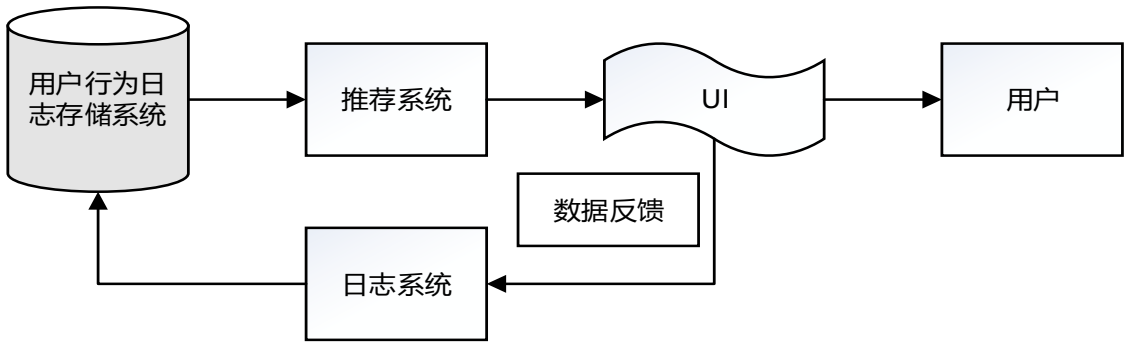


图 2.3 推荐系统一般架构

2.3 推荐算法分析

有推荐功能的系统相比普通系统当然是多了推荐功能，推荐系统的核心是推荐算法，常见的推荐算法分类[15]见图 2.4。接下来对常见的推荐算法做一简单描述。

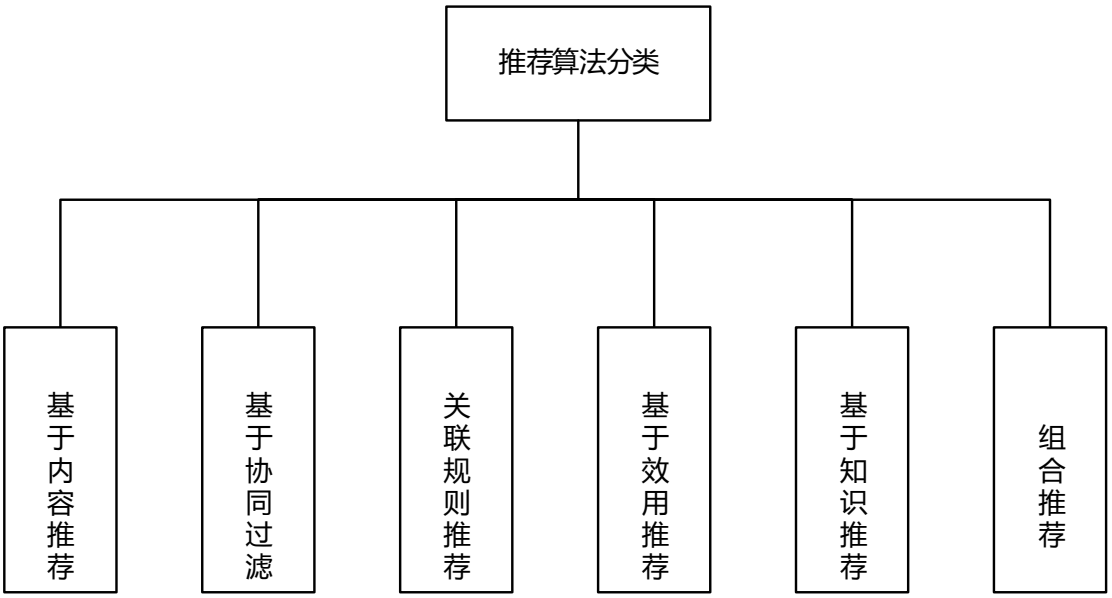


图 2.4 推荐算法分类

2.3.1 基于内容

基于内容推荐算法[16]的基本思路是，寻找用户曾经感兴趣的项目，给用户推荐与他曾经喜爱的项目类似的项目。例如：在淘宝当中，每当进入任何一个物品页面的时候都会有一个“推荐”栏目，这时候他就会根据你经常购买的物品给你推荐相似的物品。对于我经常购买电子产品，因此他会给我推荐比较相似的电子产品。

基于内容的实现主要解决两个问题。第一，确定用户的喜好，根据用户过去的浏览历史记录用 Python 的模块 Jieba 或者其他工具提取其中的关键词。第二计算两个物品的相似性，根据提取的关键词，运用 TF−IDF 算法[17]来实现推荐。此算法可以运用热点新闻的推荐，但是有冷启动的问题。

2.3.2 基于协同

协同过滤[18]是比较著名的推荐算法。协同过滤推荐算法分为两类，分别是基于用户的协同过滤算法，和基于物品的协同过滤算法。举例来说基于用户的协同过滤算法，就是推荐相似值较高的用户喜欢的项目给当前用户，它与基于内容的推荐算法区别是站在用户的角度考虑。例如，小明和小花是好朋友，某一天小明买了一本书，然后评价很好，系统会把这本书推荐给小花。

具体的实现分为两步，首先查找最近的邻居，可以运用余弦相似度公式[19]，然后计算推荐列表，缺点主要有稀疏问题和新用户问题。

2.3.3 关联规则

关联规则[20]这个算法一般用在超市或者电商系统当中，我们举例子来说明这个算法，如表 2.1 中，可以发现五条交易记录中，三条就有啤酒，尿布，因此可以构成尿布->啤酒这条规则，可以猜想购买尿布的人也许会购买啤酒。当然这些规则之间的关系需要一定的数据支撑。

表 2.1 购物清单

交易编号	商品
0	豆奶，草莓
1	草莓，尿布，啤酒，辣椒
2	豆奶，尿布，黄瓜，饼干
3	黄瓜，饼干，尿布，啤酒
4	黄瓜，啤酒，尿布，豆奶

2.3.4 组合推荐

金无足赤，人无完人。当然，算法也不列外，以上各个算法都有优缺点，在实际运用当中，可以把某几个算法相结合，组合推荐[21]的目的就是用一些算法的优点去弥补一些算法的缺点，这样可以给用户推荐更好的项目。组合的方式各种各样，例如用协同过滤推荐算法与基于内容的推荐相结合，每个方法推荐一个结果，然后可以给每一个结果赋以权重推送给用户。

2.4 本章小结

本章介绍了此系统开发的环境配置，以及用到的一些程序语言和开发工具，接着对推荐系统架构从更高一层级，更抽象的角度做了简单讲解，最后对常用的推荐算法做了分析，各种算法都有什么优缺点以及应用的场景，以便可以找到符合自己系统的推荐算法来做推荐系统。

第三章　系统需求分析和设计

3.1 系统需求分析

系统需求分析是确定我们即将开发的系统必须有哪些功能，系统的逻辑模型是在需求分析阶段确定的，它是以后设计和实现系统的基础，也是系统开发人员、开发过程和开发成本等一系列前期需求制定的详细计划。如果没有详细的需求分析，可能在以后开发过程中会出现更改数据库，更改架构甚至重新开始，这样会浪费人力财力。

3.1.1 系统结构需求分析

本系统采用当下流行的 BS 结构（Browser/Server, 浏览器/服务器结构）[22]设计，对于开发和维护来说，只要开发完成发布在服务器上，维护也在服务端维护就可以，客户端可以说是零维护。对于用户，只需要一个浏览器，就可以访问，不需要像 CS 结构（Client/Server, 客户端/服务器结构）一样需要安装客户端，这样对用户的应用提供了更便捷的方式。

3.1.2 系统功能的需求

通过对当下的视频网站和有推荐功能的网站进行分析，此类系统主要分类两个部分，前台和后台，前台供用户使用，后台供管理使用。前台是用户交互的界面，用户可以注册成为新用户，告诉系统自己的兴趣爱好，系统会根据你的爱好推荐相应的视频，这样能够化解推荐系统面临的问题之一：冷启动[23]。用户观看视频后进行评分，系统根据用户的评分以及用户浏览的历史，用这些数据不断的训练模型，推荐恰当的视频给用户。后台管理是对整个推荐系统的视频、用户和推荐算法的维护，不断的采集新视频数据，然后添加到系统，供用户观看，然后可以对一些视频信息做更改以及系统用户的维护。

3.2 系统的设计

3.2.1 系统的框架设计

生活中，盖一栋大楼，其中的人员都是分工明确，各司其职，我们的软件也是一样。把一个大的系统逐步分成为小的模块，最后分给不同的人员来完成。我们可以按照不同的维度来划分。纵向可以把整个系统的设计为三层[24]，template 层、view 层和 model 层，横向划分就是我们 3.2.2 节的模块划分。三层主要功能如下：

Model 层：和数据库中的表字段建立联系，以及供前台数据的封装传输。

Template 层：负责把从后台获取到的数据以怎样的方式展示给用户，例如表格，绘制曲线图等等。

View 层：处理业务逻辑，主要有三个部分，第一做传进来参数的校验，第二处理该功能的逻辑，第三调用数据库层获取数据。

三层之间的调用如图 3.1。

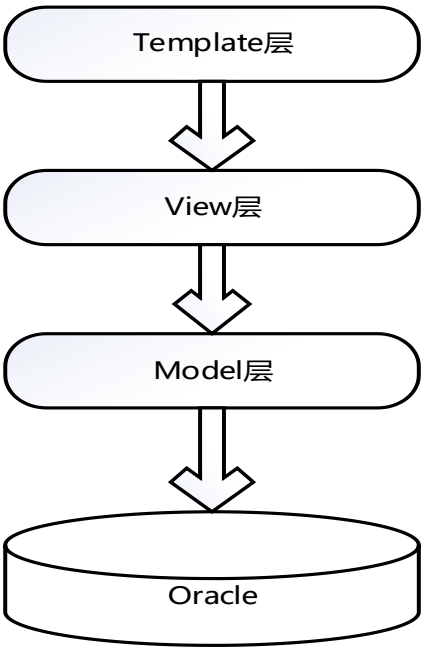


图 3.1 三层调用

3.2.2 系统模块的划分

为了更好的管理，开发，维护，本系统划分为以下几个模块，用户模块，推荐模块，日志模块，后台管理模块和定时任务模块。各个模块的功能如下：

用户模块：登录系统、没有账号则需注册、还有查找视频等功能。

推荐功能：根据用户的登录状态，查询离线计算的推荐数据，推荐符合用户的视频。

日志模块：收集用户的浏览记录，对视频的评分。

后台管理：实现对视频的管理，包括新视频的发布，修改视频的信息，还有用户管理以及算法优化，查看用户的相似值以及推荐列表，检查推荐出的视频是否符合用户的喜好。

定时任务：此系统的推荐设计为离线推荐，不论机器的环境有多好，我们的数据都在不断的增加，如果实时计算，初始阶段可能不存在性能问题，久而久之后，当用户数量增加，视频数量增加，就会存在性能瓶颈，因此，开始设计阶段就设计为离线计算。把推荐数据提前计算好，定时更新变化的数据，提高系统的性能，给用户更好的体验。图 3.2 是系统的五大模块。

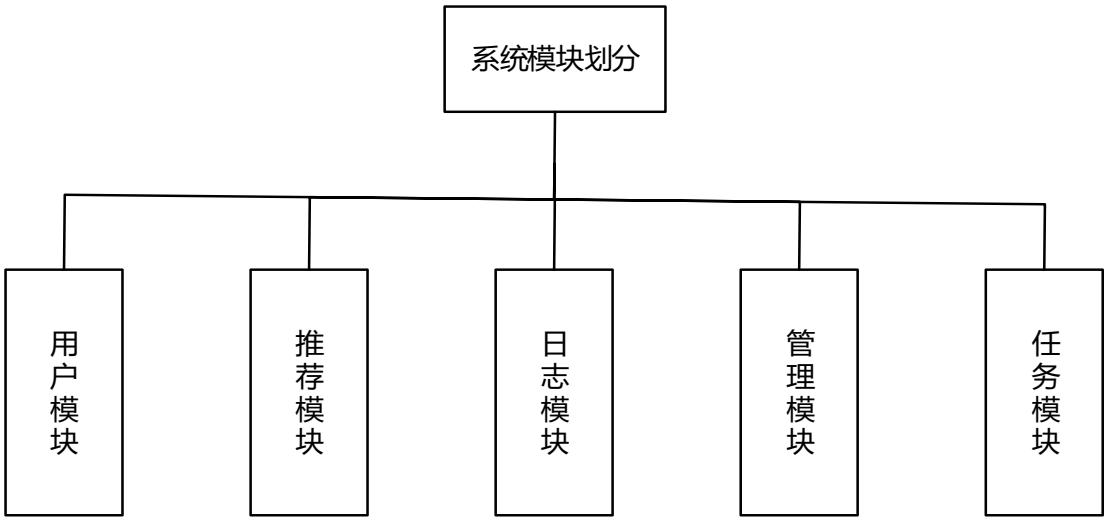


图 3.2 系统模块划分

3.2.3 用户模块的设计

前台界面为用户提供的，因此多应该站在用户的角度考虑问题。为了提升用户的体验效果，我们的系统不只是提供推荐功能，还有对视频做分类，用户可以根据自己的爱好，挑选自己喜欢的视频类型进行观看。如果用户有明确想看的视频，通过搜索功能可以更精确的找到视频，最后对观看过的视频可以做出评价，系统会根据你的评价给你推荐最符合你的视频。当然为了让系统知道你是谁，你需要登录到系统。如果你是新用户，我们还提供了注册功能，注册的时候还可以告诉系统你喜欢的视频类型，这样就更能推荐出合适的视频。用户功能如图 3.3 所示。

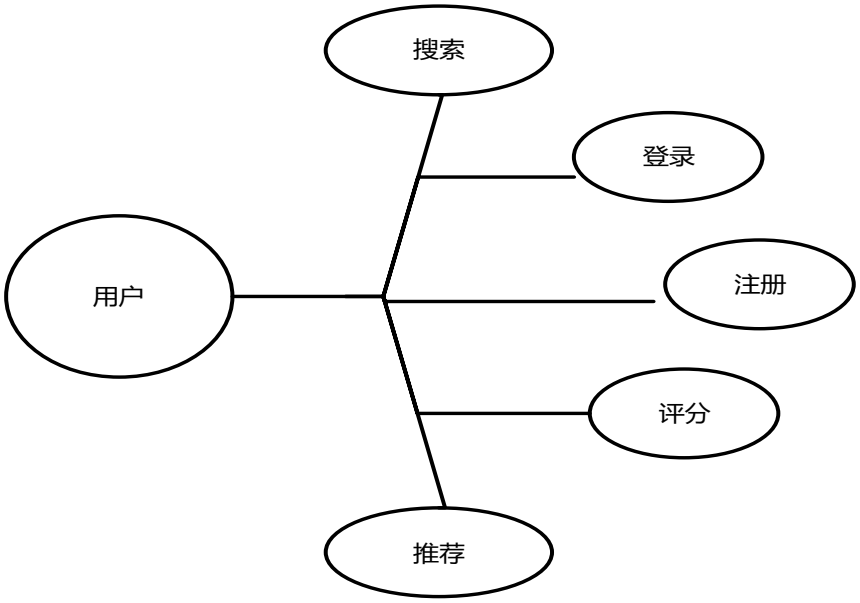


图 3.3 用户功能

3.2.4 推荐模块的设计

推荐模块当然是本系统的非常重要的模块，本系统的设计为离线计算不是实时计算，因此在推荐模块不实现推荐算法，推荐算法由定时任务调用的存储过程来实现，此模块接受用户的推荐请求，对推荐算法计算的结果查询、过滤和排序，最终推荐给用户。

本推荐系统主要使用的是基于用户的协同过滤算法做推荐功能，为了解决冷启动问题，我们结合多种推荐方式，所以在推荐模块就得做控制功能，何时该用什么算法，或者多种算法相结合实现推荐。如图 3.4 展示了我们的推荐流程。

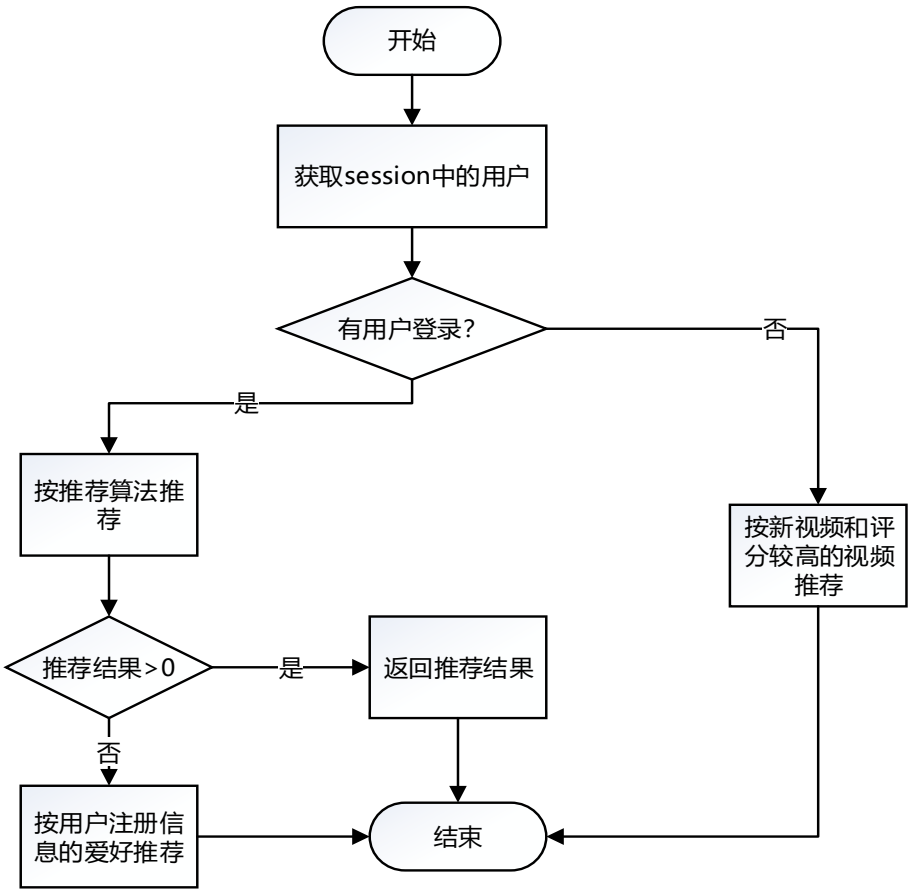


图 3.4 推荐流程

（1） 首先得到 Session 中存储的用户信息；

（2） 判断是否有用户登录；

（3） 有用户登录按照推荐算法推荐，当为新用户第一次登录的时候，推荐算法也不能推荐，此时推荐出来的数据为 0 条。这就是用户的冷启动问题，为了解决此问题我们判断是新用户的话，按照用户注册时的爱好信息，根据电影的类型给用户推荐，当用户观看一段时间后，再按照推荐算法推荐。没有用户登录，我们给用户推荐最新的视频，以及评分较高的视频。

3.2.5 日志模块的设计

推荐用户最喜欢的视频是系统的关键。那么怎么知道用户喜欢什么？这就得根据用户的浏览记录来判断用户喜欢什么，因此收集用户浏览记录就不可缺少。例如收集看过的视频，观看时间长度，类型，用户观看完后对视频的评分等信息，这就是日志模块该有的功能。

3.2.6 管理员模块的设计

如果视频的资源不更新，系统做的再好也不会有大量的用户。大量的用户中当然有非法的用户，因此对视频和用户的管理就是理所当然的事情。管理员模块主要分为视频管理、用户管理和算法优化。在视频管理中，主要做新视频的增加，修改视频的信息，顶置视频等功能。用户管理主要对不合法的用户做禁用等功能。算法优化分为相似度和推荐度，主要查看用户之间相似值以及推荐给用户某一视频的推荐值，以便我们更新优化推荐算法。图 3.5 是我们管理员模块的功能。

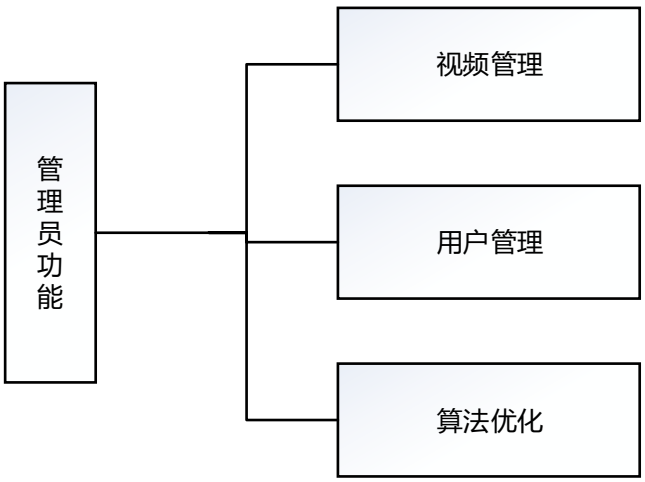


图 3.5 管理员功能

3.2.7 定时任务模块的设计

系统的设计尽可能要有扩展性，一个系统起初用户数量不是很大，视频的数量也不大，系统上线正式面向用户之后，这两者都会快速的增加。那么如果我们的推荐算法是实时计算的话，用户间的相似矩阵很大，计算时间会变长，这样推荐给用户的时间增加，用户在等待推荐结果，体检效果很不佳，因此，在设计的时候就应该考虑这一点。

为了避免以上问题的发生，我们采用离线计算推荐结果。系统初始化的时候计算好已有的数据，当有数据变化的时候，更新相应的数据。我们采用定时任务来完成。定时的检验数据是否有变化，有变化则更新变化的数据。这样可以提升系统的性能。

我们的数据都是存储在数据库，大量的计算假设使用 Python 程序语言的话，会在数据的查询和插入花费大量的时间，因此采用数据库的存储过程来计算，减少连接数据库的开销时间。推荐算法采用基于用户的协同过滤算法，主要分两步来实现：

（1） 寻找和将要给推荐的用户喜好相近的所有用户。

（2） 遍历这个用户集合，寻找该集合中用户喜爱的项目，并且将要给推荐的用户过去没有对该项目有过“喜爱”标签的项目推送给用户。

在第一步当中主要是求每个用户与其他用户的喜好值。假如有两个用户 m 和 n，令 U(m)为 m 用户过去喜爱的项目集合，令 U(n)为 n 用户过去喜爱的项目集合。通过余弦相似度公式 3.1 计算：

$$W_{mn}=\frac{|U(m)\cap U(n)|}{\sqrt{|U(m)||U(n)|}} \tag{3.1}$$

下面以用户喜欢的行为记录为例, 如图 3.6，举例明 UserCF 计算用户兴趣相似度的例子：

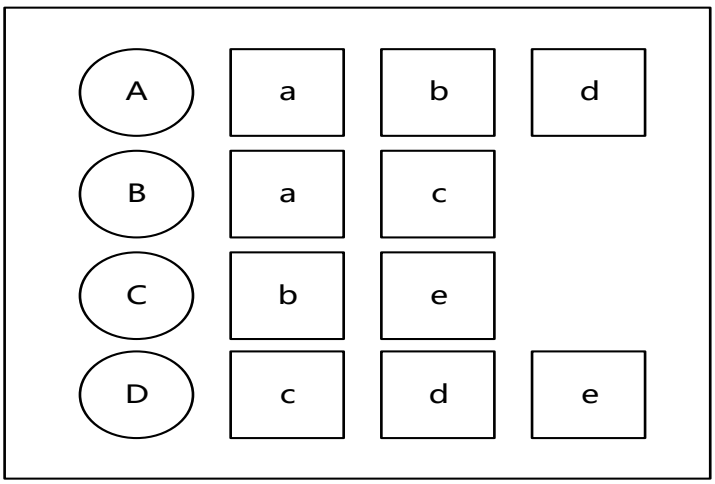


图 3.6 用户物品图

利用余弦相似度公式计算用户 F 和用户 G 的喜好相似值为

$$W_{FG}=\frac{|\{f,g,j\}\cap\{f,h\}|}{\sqrt{|\{f,g,j\}||\{f,h\}|}}=\frac{1}{\sqrt{6}} \tag{3.2}$$

同理，我们可以计算出用户 F 和用户 H、J 的相似值：

$$W_{FH}=\frac{|\{f,g,j\}\cap\{g,l\}|}{\sqrt{|\{f,g,j\}||\{g,l\}|}}=\frac{1}{\sqrt{6}} \tag{3.3}$$

$$W_{FJ}=\frac{|\{f,g,j\}\cap\{h,g,l\}|}{\sqrt{|\{f,g,j\}||\{h,g,l\}|}}=\frac{1}{3} \tag{3.4}$$

根据公式 3.1 我们可以获取两个用户间的喜好相似值，基于用户的协同过滤算法会给用户推荐和他喜好非常相近的 T 个用户喜爱的项目，用公式 3.5 度量基于用户协同过滤算法中用户 m 对物品 r 的喜好度：

$$p(m,r)=\sum_{n\in C(m,T)\cap M(r)} S_{mn} \qquad (3.5)$$

在公式 3.5 中， C(m,T)表示和 m 用户喜好非常相似的 T 个用户，集合 M(r)是对项目 r 有过浏览的用户集合，S_{mn} 是 m 用户和 n 用户的喜好相似值。

3.3 数据库的设计

上一节当中对系统做了详细的设计，每个模块都有什么功能做了详细的阐述，对推荐流程给出了流程图，推荐算法给出了案例。这些功能的数据都要存储起来，要么存储在文件，要么存储在数据库。数据库的速度快，读写方式也比较简便，因此我们采用数据库。

3.3.1 表的详细设计

更好的能体现推荐功能，需要用户登录到系统。为了存储用户的信息，我们设计表 3.1 用户表。有 id 来标识唯一用户，以及注册的时候收集用户的爱好信息。密码和用户名为了登录到系统当中，状态来标识你是否为一个合法的用户。各个字段的数据类型以及名称等信息见表 3.1。

表 3.1 用户表

字段名	数据类型	长度	允许空值	备注
ID	NUMBER	-	否	用户 id
USER_NAME	VARCHAR2	255	是	用户名称
PASSWORD	VARCHAR2	255	是	密码
EMAIL	VARCHAR2	255	是	电子邮件
HOBBY	VARCHAR2	255	是	爱好
STATE	VARCHAR2	1	是	状态

视频是我们推荐的数据，每个视频都有一个 id 唯一标识，用户需要了解了视频的信息后才选择是否观看，因此视频应该有题目，年份，属于那种类型，以及视频的简介信息，为了观看还应该有播放的地址，为了顶置一个视频，我们用一个字段标识该视频是否为新视频。表 3.2 是视频表的具体设计。

表 3.2 视频表

字段名	数据类型	长度	允许空值	备注
MOVIE_ID	NUMBER	-	否	电影 id
TITLE	VARCHAR2	255	否	标题
YEAR	DATE	-	是	年份
GENRES	VARCHAR2	255	是	分类，以|分割
DESC	CLOB	-	是	描述
LINK	VARCHAR2	255	是	链接
ISNEW	NUMBER	-	是	是否为新视频
DIRECTOR	VARCHAR2	255	是	导演
LEADACTORS	VARCHAR2	255	是	演员，以, 分割
PICTURE	VARCHAR2	255	是	图片

训练合适的模型，给用户推荐数据，我们需要收集用户观看后对视频的评分信息。评分表见表 3.3。本系统的所有用户都能对推荐的每个视频打分，因此应该有用户信息，视频信息和评分，当用户给一个视频评分以后，用户的推荐数据就有可能变化，因此我们设计一个字段，来标识是否需要更新，定时任务来检查该表，来更新推荐数据。

表 3.3 评分表

字段名	数据类型	长度	允许空值	备注
USER_ID	NUMBER	-	否	用户 id
MOVIE_ID	NUMBER	-	否	视频 id
RATING	NUMBER	-	是	评分
ISUPDATE	VARCHAR2	1	是	是否更新
TIMESTAMP	VARCHAR2	100	是	时间戳

推荐算法的实现分为两步，首先遍历用户计算每个用户与其他用户对应的相似值，第二步计算推荐每个视频的推荐值。因此相似用户表设计为两个用户和两个用户的相似值，见表 3.4。

表 3.4 相似用户表

字段名	数据类型	长度	允许空值	备注
USER_ID1	NUMBER	–	否	用户 1
USER_ID2	NUMBER	–	否	用户 2
VALUE	NUMBER	–	否	相似值

根据相似用户计算推荐值，因此有用户信息，视频的信息，以及推荐值，用户我们用用户 id 表示，视频用视频 id 表示。具体的见表 3.5。

表 3.5 推荐列表

字段名	数据类型	数据类型	允许空值	备注
USER_ID	NUMBER	-	否	用户 id
MOVIE_ID	NUMBER	-	否	视频 id
RECOMMEND_VALUE	NUMBER	-	否	推荐值

3.3.2 E-R 图

E-R 图是现实世界的概念模型，能更清楚的表示每个实体的属性以及各个表之间的关系。图 3.7 描述了本系统中主要的三张表（评分表，视频表和用户表）的 E-R 模型。

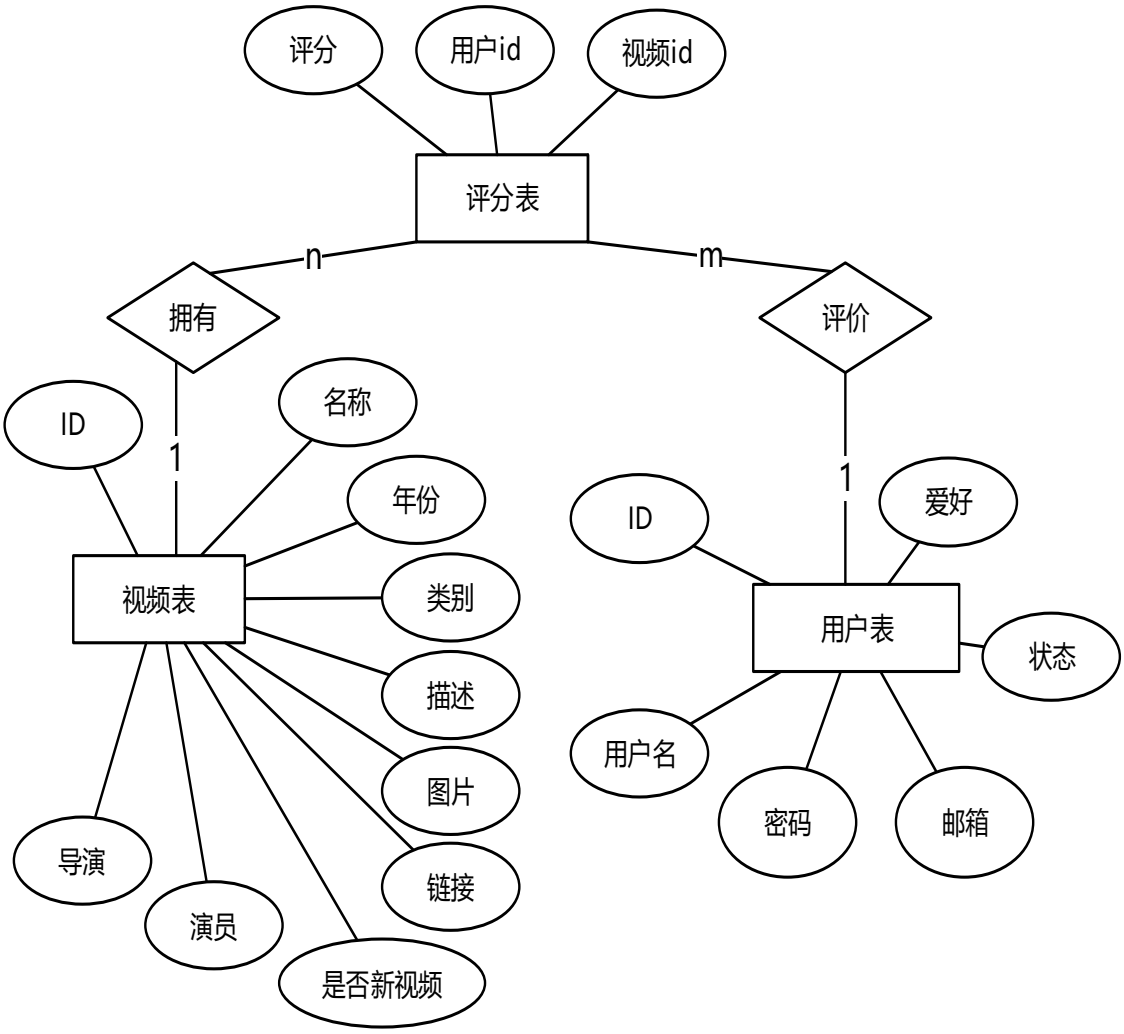


图 3.7 E-R 模型

3.4 本章小结

本章从软件工程的角度来讲解，主要包括系统的需求分析，详细设计，确定了个性化视频推荐系统该有那些功能，对推荐流程给出了详细的流程图，以及推荐算法用案例来讲解，明确了系统的开发方向。最后对需求分析和设计阶段提出的功能的数据存储做了讲解，对每个模块需要的表和字段以及之间的关系做了详细设计。做好设计是开发软件的前提。

第四章　视频推荐系统的实现

4.1 系统实现

在上一章中，对系统做了需求分析和设计，清楚了系统该有那些功能，系统的实现有了基本的方向。本章中，从实现方面来做详细叙述。个性化视频推荐系统采用 Python 程序语言开发，利用 Django Web 框架，前台页面主要用 HTML 显示内容 ，CSS 来做美化，JavaScript 实现动态的效果以及 BootStrap 等前端框架。数据库采用常用的关系型 Oracle 数据库。此章主要讲在本推荐系统开发过程的技术问题，实现逻辑和主要的流程。图 4.1 是各个功能之间通过共享数据库之间的联系。

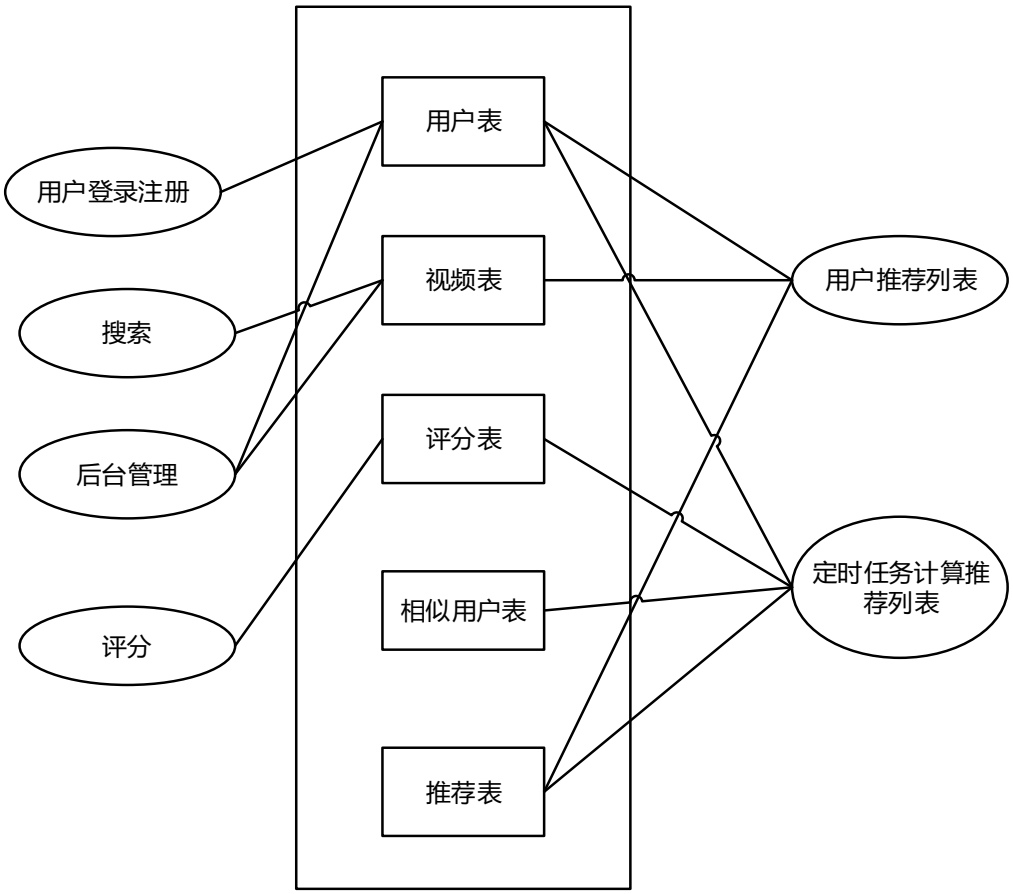


图 4.1 各个功能之间的联系

4.1.1 Model 层的开发

在本层当中主要对数据库中表对象的增加、删除、修改和查询等操作，本层在整个系统中的作用如图 4.2。

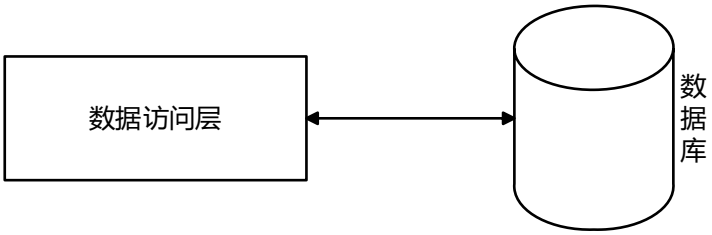


图 4.2 Model 层的作用

在这一层主要封装了对数据库的访问细节，在原始的开发当中，我们如果想在计算机语言里面执行 SQL 语言，我们平常都进行如下的操作：

（1） 建 Database，创建表和添加字段；

（2） 使用 cx_Oracle 驱动文件（如果用 oracle 数据库）来连接数据库，并编写数据访问层代码（获取连接，编写 sql 语句，执行 sql，关闭连接）；

（3） 业务逻辑层去调用数据访问层执行数据库操作。

在本系统当中，我们运用 Django 为我们提供的访问数据库的框架。类和表之间的对应关系如图 4.3。在访问数据库时，只需要用对应的类，调用恰当的 API 就可以完成。减少代码的编写提高开发效率以及后期的维护。这也是我们选择 Django 框架的主要原因。

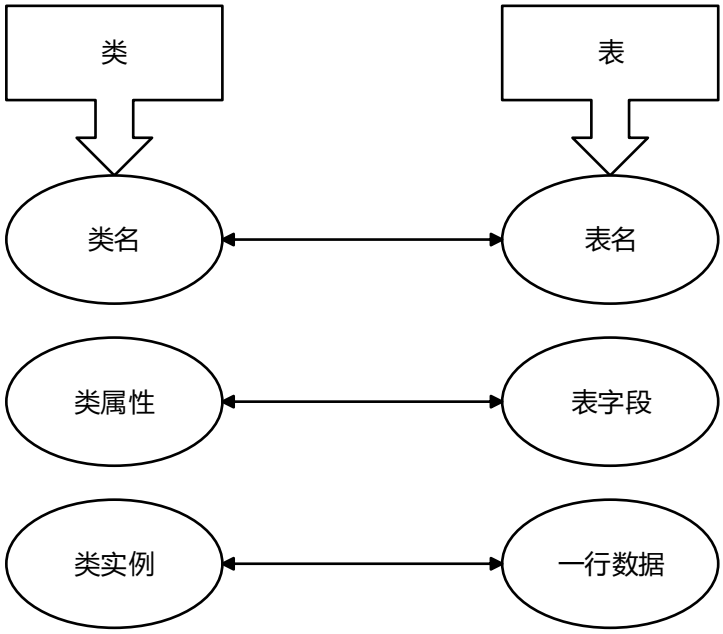


图 4.3 类到表的映射

那么，我们的表和类怎么关联起来呢？分为两种情况：第一：根据类创建表，第二根据表创建类。不管是哪一种，查看 Django 提供的文档，只要执行相应的语句就可以让表跟类关联起来，然后操作相应的 API。

4.1.2 View 层的开发

用户请求过来的主要逻辑实现就在这层，位于其他的两层之间，对实体的增删改查操作。

业务逻辑层是三层中非常重要的一层。主要包含业务逻辑的流程，数据合法性的校验规则，与数据库连接的实体对象，展现给前台的对象模型。数据合法性校验：业务层有三分之一部分要做数据验证，例如调用业务接口时传过来的参数校验，以及从数据库层查找出来的数据校验等等。业务逻辑的流程：业务层的其他三分之二就是处理逻辑流程，实现这个功能应有那些算法，这是开发过程当中灵活度比较高的模块，我们需要大量的 IF-ELSE 或 SWITCH-CASE，我们可以抽象出这里的逻辑作为一个私有子服务专门使用。与数据库连接的实体和显示到页面的数据有时间不能共用，因此出现数据库连接的实体和展现给前台的对象模型。

如图 4.4 是业务逻辑层在软件结构中的位置：

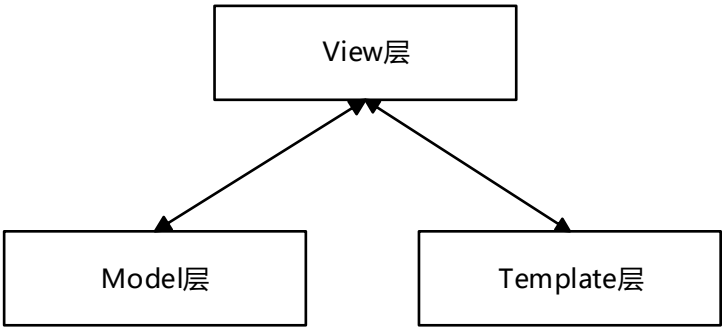


图 4.4 业务逻辑层的位置

4.1.3 Template 层的开发

传统上，我们前台和后台有紧密的联系，前台开发人员要关注后台，后台开发人员要关注前台，不能达到各司其职。如果能把前台和后台的耦合度减小，这样减少开发人员的对其他方面的关注度。所以出现了模板的开发。开发模板层时主要包括 html+模板语法，我们把需要的数据用变量替换，再加一些逻辑判断即可。

例如：在用户模块，我们利用 IF ELSE 逻辑标签检查用户是否登录，如登录使用 Session 中的用户替换模板，若没有，显示登录按钮让用户去登录。

4.2 功能实现

本小节讲解各个功能的具体的实现。在开发过程中，前台的请求以 Ajax 的方式请求，后台以 Json 的格式返回数据。Json 的格式信息规范如表 4.1：

表 4.1 返回数据类型规范

code	info	data
四位数字代码	注册成功	User
{“code”：“0000”，“info”：“注册成功”，“data”：user}		

4.2.1 用户模块的实现

1. 用户注册

图 4.5 用户注册界面

想要给用户带来更好的推荐体验，推荐符合用户的视频，需要让系统知道你是谁，如果是老用户已经有账号，直接登录。如果没有账号就是新用户，新用户需要先注册才能登录，注册页面如图 4.5。需要填写用户名、密码、电子邮件和选择你喜欢的视频类型。图 4.6 是新用户注册功能的处理流程。

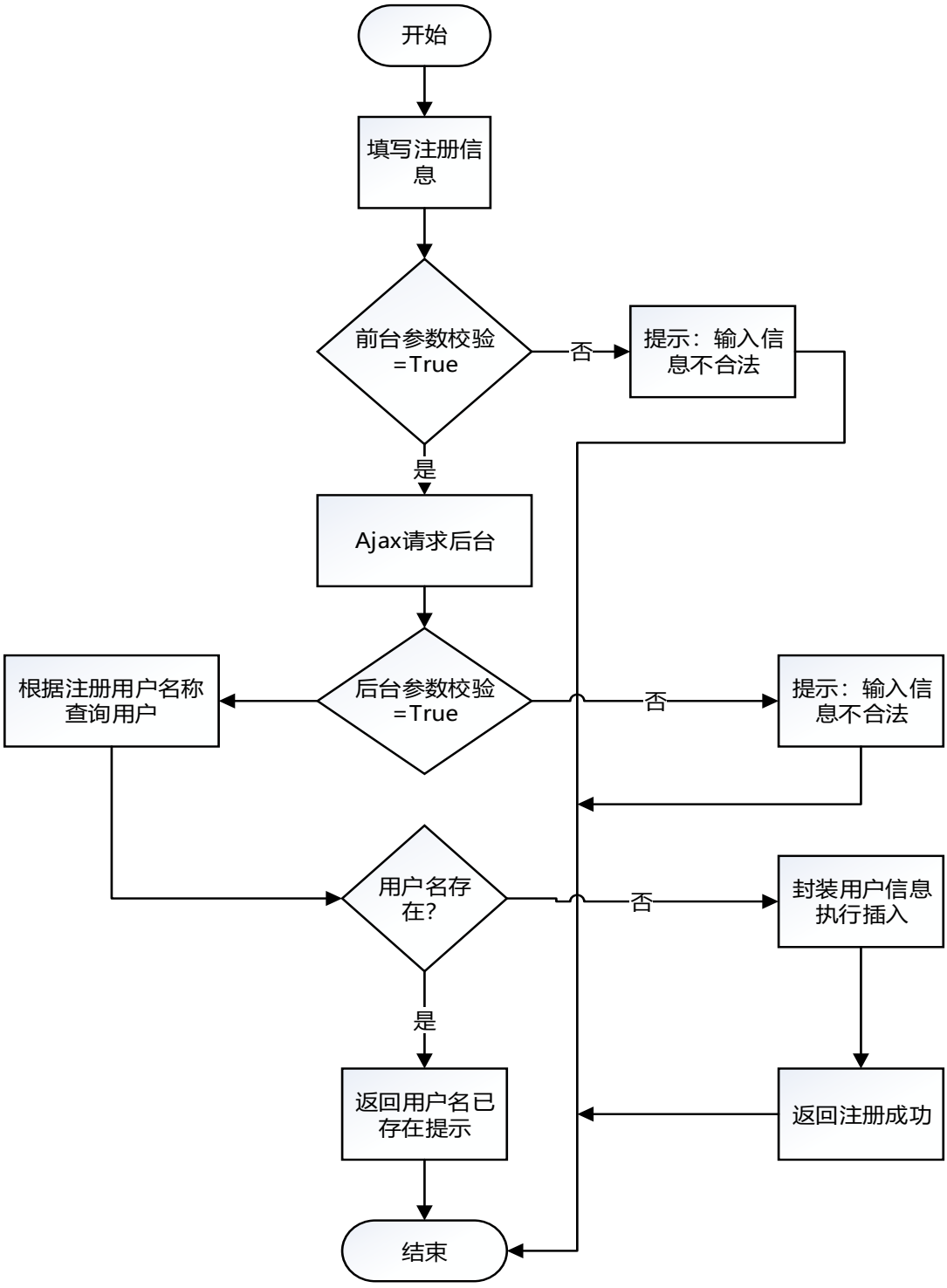


图 4.6 用户注册流程图

在系统的首页有登录->注册的按钮，点击进去之后填写相应的信息。填写完提交之后前后台都会做相应的校验工作，然后判断用户名是否已注册，未注册则给用户生成 ID（此 ID 使用 Oracle 的 Sequence），把对应的其他信息插入到数据库，则注册成功，用户可以去登录页面登录。

2. 用户登录

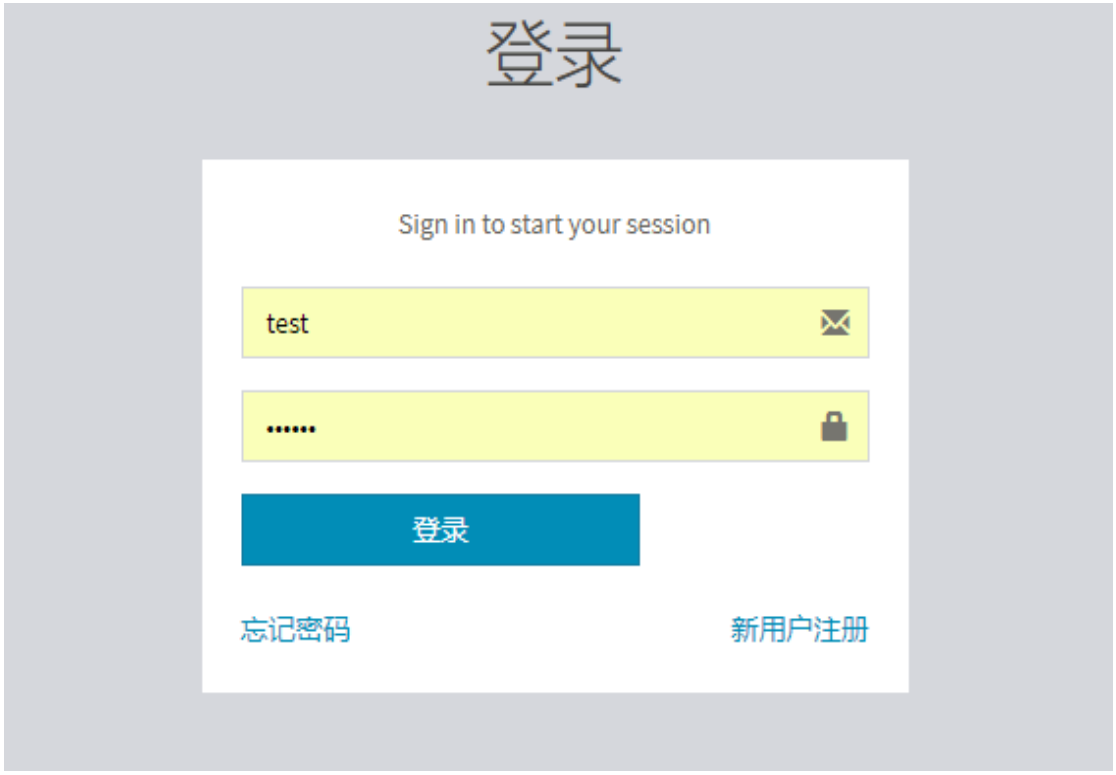


图 4.7 用户登录界面

用户登录的界面如图 4.7，有密码和用户名两个输入框需要用户填写。用户登录的具体流程为：当用户提交登录信息后，前台用 JS 对用户填写的数据做校验，用户名和密码都不能为空，密码不能低于六位。在前台校验成功后，然后封装参数以 ajax 的方式请求到后台，这样可以减少服务器的压力。后台获取登录的信息后，也必须校验。在某些情况下，用户可以不通过前台页面直接请求到后台，因此后台的校验也是必须的。校验通过后再从数据库查找，判断该用户是否存在以及用户名密码是否正确。若存在并且正确，则把用户信息放入 Session 当中，供前台页面显示。否则返回错误消息到前台页面，然后展示给用户。登录流程如图 4.8。

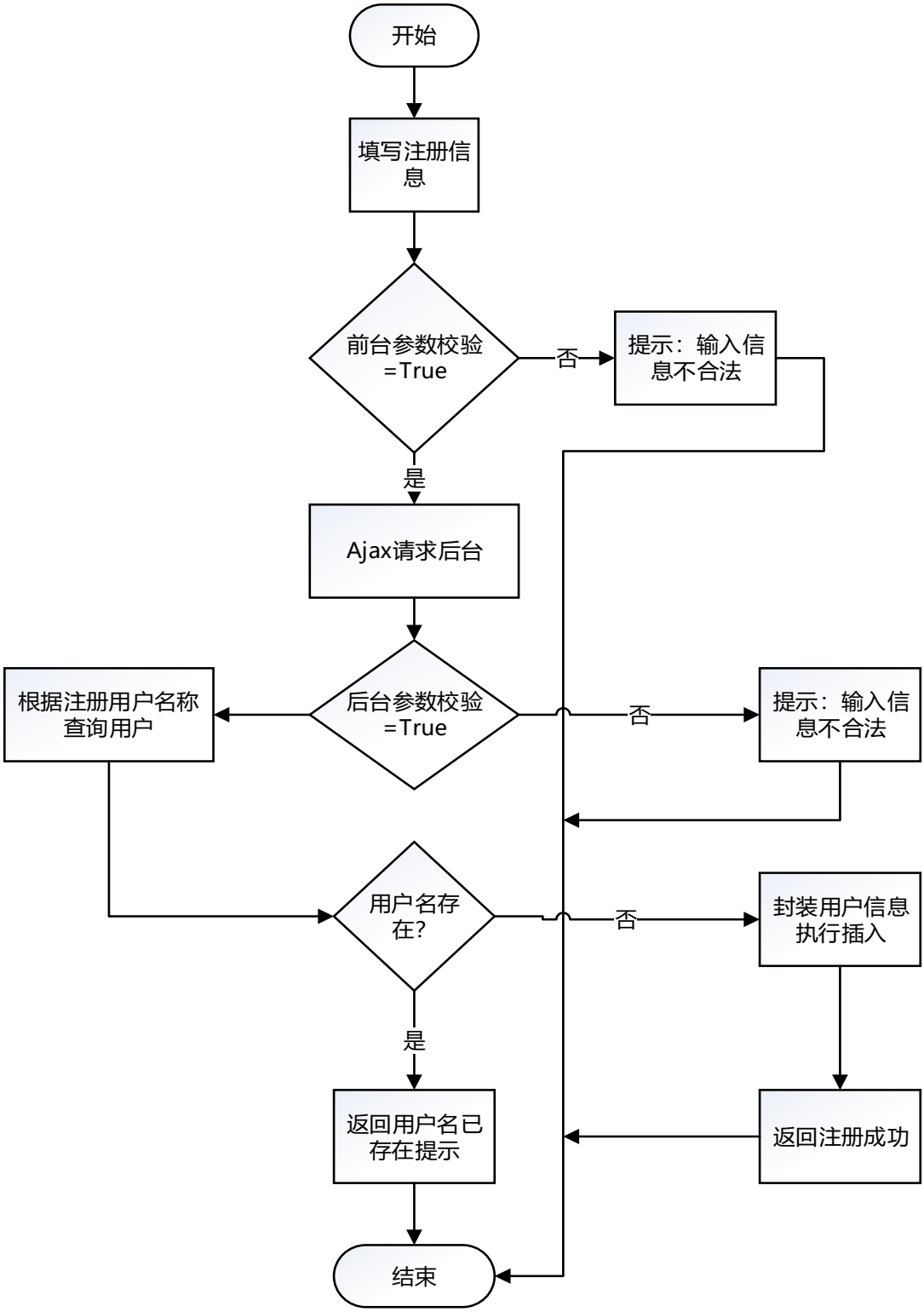


图 4.8 用户登录流程图

用户登录成功后的主页面如图 4.9。左半小部分会列出推荐给当前登录用户的十条记录，展示缩略图和视频的部分信息以及推荐理由，中间为播放视频，下边用户可以对视频进行评分。右边显示当前视频的具体信息。

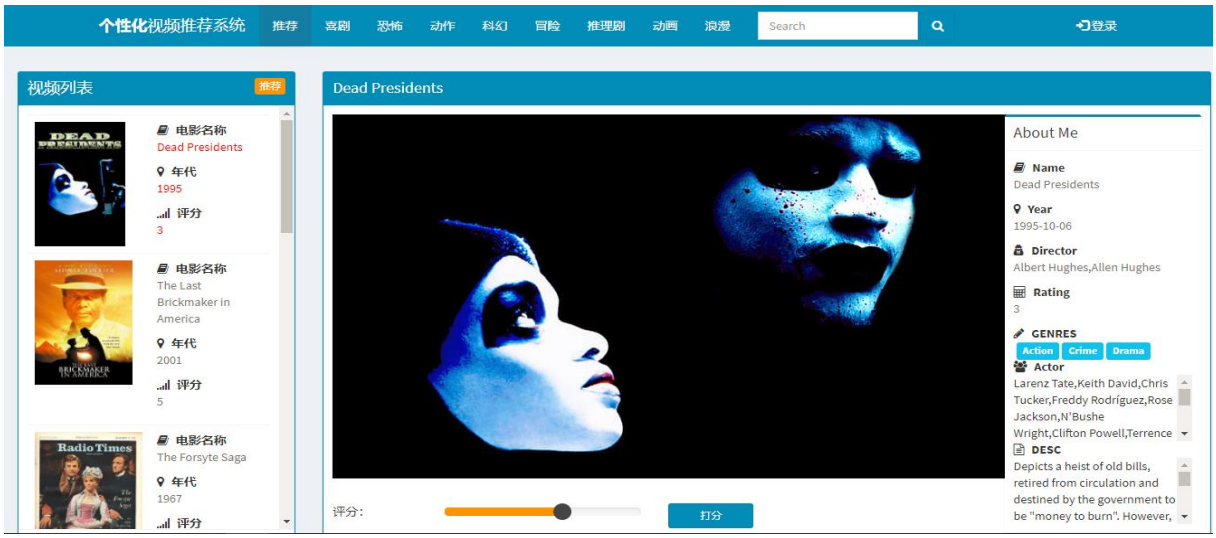


图 4.9 用户主界面

用户有登录操作相对应的就有退出操作，用户退出主要是销毁 Session 中用户的信息，界面显示把用户名称更换为登录的按钮。前台点击退出按钮后发送 ajax 请求到后台，后台返回成功后前台重新加载页面，此时 Session 中已经没有用户的信息，因此在界面中显示登录的按钮。

3. 用户其他操作

界面菜单如图 4.10。用户在界面可以进行查找，在搜索框输入关键词后，点击查找按钮之后，系统会根据输入的关键词按视频的名称进行模糊匹配。然后将查到的视频返回给用户。

图 4.10 界面菜单

界面中还对视频进行分类，在导航栏中有推荐菜单和视频的分类。用户点击某一个分类后后台根据类型会为用户查找出相应的视频，为了解决个性化视频推荐系统中新视频的冷启动，如果查出来的视频中含新视频，我们会把新视频顶置，以便用户可以看到新视频。此处都是用户查找操作，因此我们把按分类和用户的搜索抽象成一段公用的逻辑，符合软件设计的规范。

4.2.2 推荐模块的实现

推荐模块是我们个性化视频推荐系统的重要模块。根据我们第三章的设计，推荐算法的实现交给我们的定时任务模块调用的存储过程来实现。因此推荐模块相对简单。如果用户没有登录，我们的推荐结果如图 4.11，推荐新视频以及评分比较高的视频，新视频会顶置并且视频名称标注为红色，如果用户登录了，并且为新用户没有观看记录，则

推荐与用户兴趣爱好相同的视频，如果用户有观看记录，则按照我们的推荐算法来推荐，如图 4.12，每个视频给出了推荐系数，这样能给让用户更加信任推荐的结果。

图 4.11 无登录推荐结果

图 4.12 登录推荐结果

4.2.3 日志模块的实现

日志模块主要记录用户看过视频的评分信息，此系统的推荐主要也依赖评分信息。如果用户是第一次给某电影评分，则插入一条记录，此种情况会发生在分类的视频或者查找的视频或者推荐的视频，推荐的视频都是用户未评分的。如果插入的记录已经存在，我们会更新原来的评分。此种情况发生在分类视频和查找的视频当中，因为其中可能有用户看过的视频。因此推荐的类型就掌握在用户的手中，你可以修改评分从而改变推荐结果。

4.2.4 管理模块的实现

管理模块主要对视频和用户管理以及算法优化，管理的界面如图 4.13。后台管理的页面分为左右两个部分，左边是菜单，右边是我们数据的展示。左边的菜单有视频管理，

用户管理和算法优化。我们先看视频管理，在右边会列出所有的视频，我们会把新视频顶置，一行代表一个视频，在每行的最右边有一个操作的栏目，可以对此视频进行删除操作和修改操作。在最上面有一个添加的按钮，可以添加新的视频。

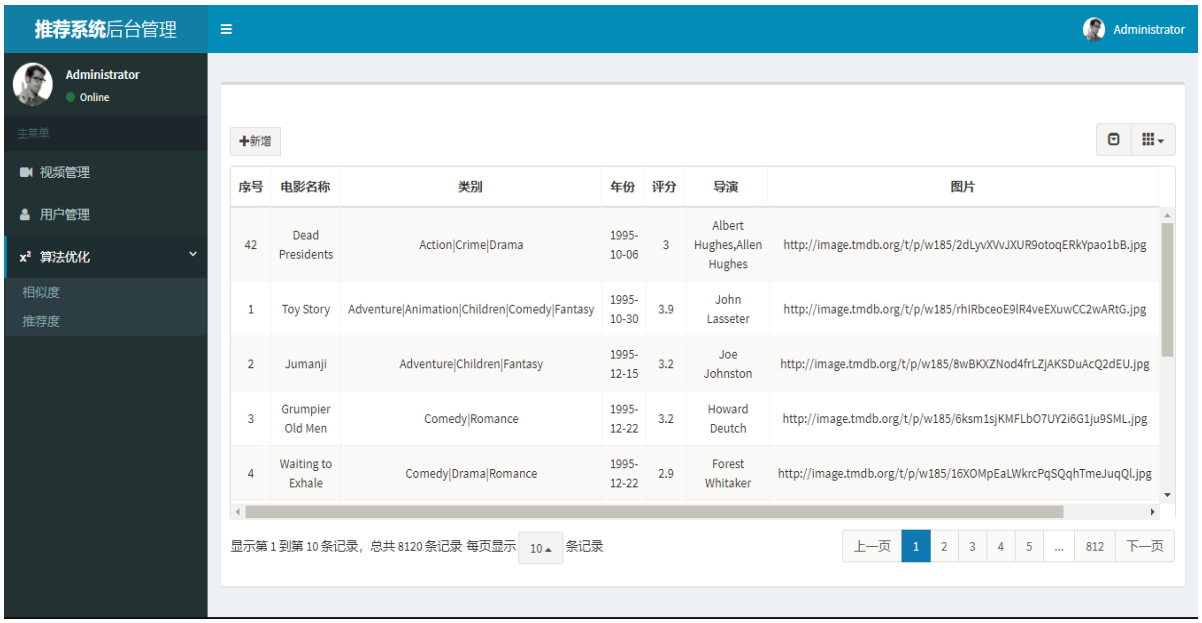


图 4.13 视频管理后台界面

视频删除：点击删除的时候会给出如图 4.14 的确认框， 确实是否删除此视频，如果确认则向后台传视频的 ID 进行删除，取消则不删除。

图 4.14 删除操作确认框

视频修改：点击修改的图标回显此视频的信息如图 4.15。此时可以选择关闭或者更改信息后点击修改按钮。前台把页面的数据传给后台进行修改。后台获取参数，进行校验，校验合法然后修改。

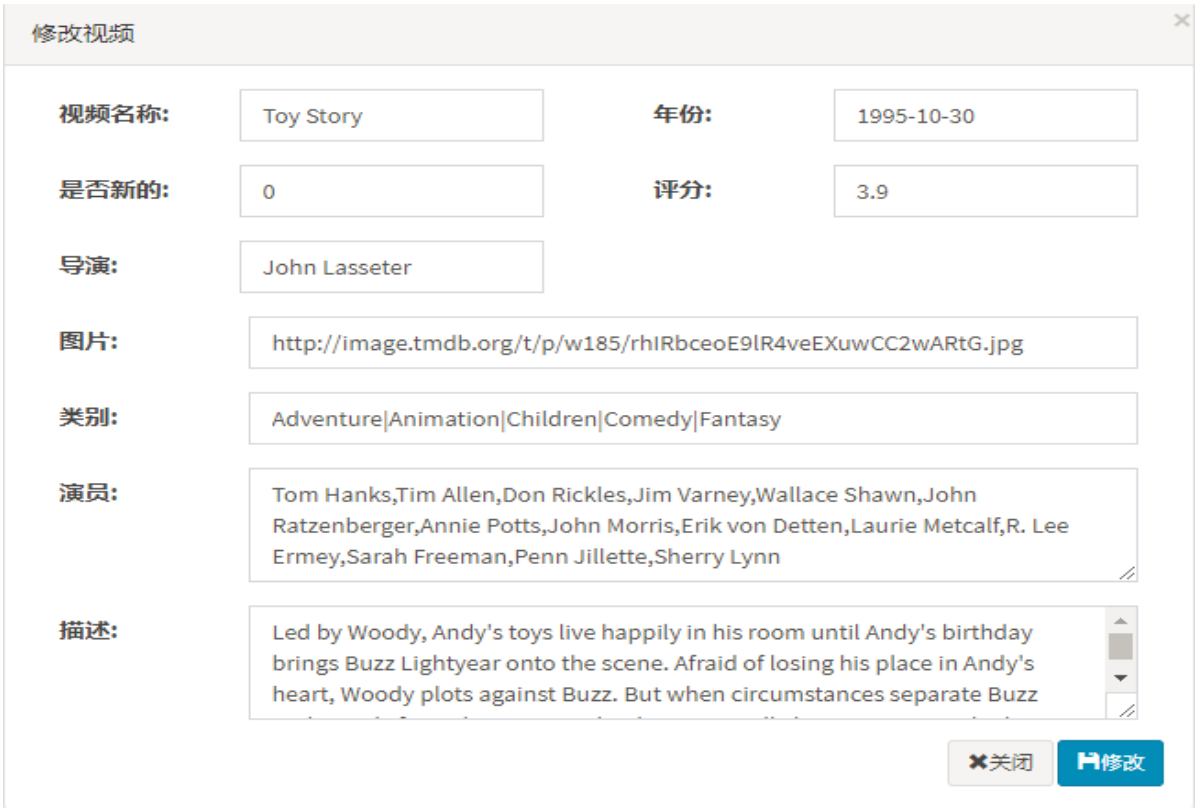


图 4.15 修改视频界面

视频增加：点击添加按钮之后会有一个弹出框如图 4.16 来填写视频的信息。填写完成后点击保存即可增加新视频：后台也是获取参数与校验操作，与更新不同的是，这里先查找到视频库最大的视频的 ID，然后给此 ID+1 作为新视频的 ID。

图 4.16 增加视频界面

图 4.17 是用户管理的界面。用户管理比较简单，主要对非法的用户进行禁用，在用户列表的最右边的操作中有禁用的按钮，点击此按钮就可禁用此用户，禁用后用户不可以登录到此系统。

图 4.17 用户管理界面

4.2.5 定时任务模块的实现

1. 算法实现

在实现推荐算法的过程中考虑到每次都要从数据库获取到数据，再计算，最后存储到数据库，这样访问数据库有大量的系统资源和时间的消耗，为了解决此问题，我们不运用 Python 语言来做推荐系统算法的开发，而是采用 Oracle 的存储过程。这样大大的提升了计算速度。

相似矩阵的计算(计算流程如图 4.18)：首先删除数据库中要计算的用户原来的相似矩阵，其次遍历所有用户计算每个用户与其他用户的相似值。

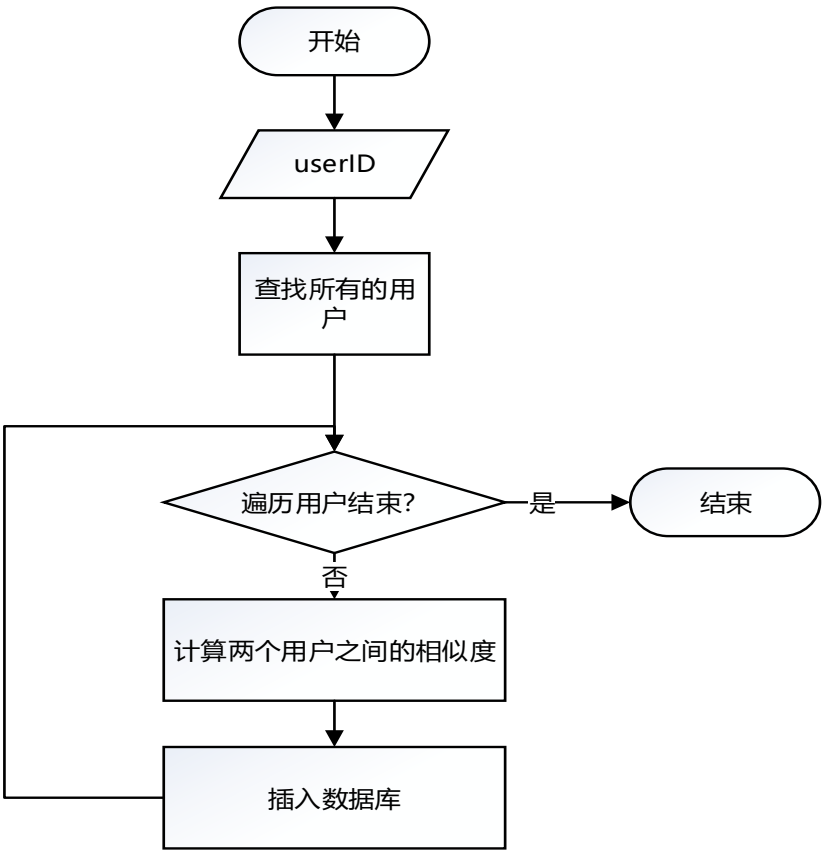


图4.18 相似用户计算流程图

2. 定时任务实现

基于 Python 语言的定时调度框架有 APScheduler 和 Celery。我们推荐系统定时任务的实现采用 APScheduler 框架，它提供了多种不同的调度器和存储器，Quartz 里的提供的每个功能在 APScheduler 里都有。对开发人员提供了便捷的操作。

我们的定时任务不断的判断是否有更新的推荐数据，当用户对某个视频的评分改变之后，这条记录在数据库表示为需要更新，定时任务查询到后调用存储过程来更新该用户的推荐列表。最后更新该条记录为不再需要更新的状态。

定时任务还需要实现另一个功能，当新视频上架之后，什么情况下不是新视频。这个功能我们也用定时任务来完成，例如规定当超过 50 个用户观看之后取消新视频的标志。

4.3 本章小结

本章从系统的实现方面进行了讲解，系统每个模块的开发都采用 MTV 的结构进行开发，对复杂的逻辑给出了流程图，每个实现还有文字描述，对开发出的界面都有截图展示。至此，本系统开发阶段完成。

第五章　个性化视频推荐系统测试

到目前为止，我们从需求分析，设计，实现方面已经做了讲解，个性化视频推荐系统已经初步完成。但是为了此系统的正常使用，需要对系统进行测试，这也是软件开发过程中必不可少的一个环节。针对系统的特点和实际情况，本系统采用手工的方式对系统测试和验收，测试的内容主要为单元测试和兼容性测试。

5.1 单元测试

为了保证划分的各个模块能够协同工作，给用户提供准确适合用户的推荐服务，对各个模块进行单元测试。设计到界面中每一个可以操作的元素。这些被测试的方法包含用户前台操作的模块，系统推荐的模块和定时任务更新推荐数据的模块。

5.1.1 用户界面测试

用户界面测试的基本流程为：对系统页面中的按钮，链接，输入框和搜索框进行点击测试，看能否点击输入等操作。具体的方案如表 5.1：

表 5.1 用户界面测试

序号	位置	测试用例	预期结果
1	主页	点击登录按钮	弹出登录框
2	登录框	点击注册按钮	弹出注册框
3	登录框	点击登录提交按钮	登录框消失，显示主页
4	注册框	点击注册提交按钮	注册框消失，显示登录框
5	主页视频列表	点击视频名称	右边显示视频详细信息
6	主页导航栏	点击搜索按钮	主页视频列表显示搜索的视频

5.1.2 系统基本功能测试

老用户可以直接登录，当然新用户先注册在登录，登录成功后显示用户基本信息，用户在观看视频之后可以对视频做出评价打分，推荐引擎根据你的评分给你推荐相应的视频。系统还拥有检索功能，用户能用关键词进行搜索，方便用户查找。表 5.2 展示了系统基本功能测试的 case。

表 5.2 系统基本功能测试

序号	位置	测试用例	预期结果
1	登录框	输入正确的用户名和密码	登录框消失，主页显示用户的基本信息
2	登录框	输入不正确的用户名或密码	弹出提示信息，用户重新填写信息再登录
3	注册框	填写注册信息并提交	校验成功后弹出登录框登录
4	主页视频详情	点击评分按钮	评分成功给出提示
5	主页视频详情	点击播放按钮	播放视频
6	主页视频列表	点击更多按钮	展示当前分类的下一页视频
7	主页视频列表	点击推荐按钮	推荐引擎推荐更多视频

5.1.3 系统推荐功能测试

推荐功能测试主要测试推荐引擎是否按照之前设计的推荐逻辑来推荐，测试 case 如表 5.3。

表 5.3 系统推荐功能测试

序号	位置	测试用例	预期结果
1	主页->导航->推荐	用户没有登录	推荐新视频和评分较高的视频
2	主页->导航->推荐	用户登录没有观看记录	按照用户注册时的爱好推荐
3	主页->导航->推荐	用户登录有观看记录	按照推荐算法推荐

5.1.4 管理功能测试

管理功能测试主要对视频的增删改功能进行测试，确保管理员在后台可以正常的维护系统，每个功能的测试步骤在表 5.4 中已经列出。

表 5.4 管理功能测试

序号	位置	测试用例	预期结果
1	管理页面->视频管理	点击删除按钮	弹出确认框确认是否删除
2	管理页面->视频管理	点击修改按钮	弹出对话框回显该视频的详细信息
3	管理页面->视频管理	点击增加按钮	弹出对话框，等待用户填写视频信息
4	管理页面->用户管理	点击禁用按钮	禁用当前用户
5	管理页面->算法优化	点击推荐度	显示用户推荐图表
6	管理页面->算法优化	点击相似度	显示用户相似图表

5.2 兼容性测试

Bs 系统的架构是浏览器/服务器结构，对用户来说就只要一个浏览器就可以使用该系统，但是，考虑到每个用户所使用的浏览器可能会不同，本小节对系统的兼容性测试。目前常用的浏览器有 Google 和火狐，通过这两个浏览器的测试，页面中的元素能正常显示并且各个功能都能正常使用，因此，兼容性测试通过。

5.3 本章小结

本章对个性化视频推荐系统开发的每个功能进行了检验和测试，所有的功能都符合期望的效果。考虑到用户所用的浏览器可能不同，因此对兼容性进行了测试。到此，本系统开发完成。

总结与展望

个性化视频推荐系统已经按最初的设计开发完成，经过测试各个功能都能正常运行，能个性化的为用户推荐视频。但因为各种因素的限制，在设计和考虑问题方面还有很多不完善，现将系统已有的功能以及个人觉得需要改善的地方总结如下。

此系统采用 Python 语言，Django web 框架在 Pycharm 编辑器中开发，用户界面使用 Bootstrap 框架，以及 JavaScript 做动态的交互，把产生以及用到的数据存储在 Oracle 数据库。

系统的设计纵向划分为三层，模型、模板和视图三层，这样对实现和今后的维护都提供了很大的便利。

系统允许用户注册为新用户，根据类别查看自己感兴趣的视频，然后对视频做出评分。本系统也给用户推荐恰当的视频，当然用户可以在搜索框中搜索具体的视频来观看。在推荐的时候采用基于用户的协同过滤算法给用户推荐。为了克服冷启动问题，我们注册的时候收集用户的爱好，解决此问题，在视频分类当中，顶置最新的视频，来克服新视频的冷启动问题。后台提供方便的界面操作，对视频和用户进行管理。

由于个人能力和时间有限，有些功能点不够完善，还需要进一步的设计完善。在界面当中，因为收集的数据的限制，不能播放视频，只展现了视频的缩略图、名称以及推荐的理由，不能给用户一个更好的体验。在推荐算法方面，只采用了单一的基于用户的协同过滤算法，收集的用户日志也只有评分信息，因此在推荐算法方面还有很多需要改进的。采集了大约 8000 多条视频数据，系统有 600 多个用户，个人笔记本配置的限制，稍微有点性能问题，以上问题都需要今后不断的改进和完善。

参考文献

[1] 王国霞，刘贺平．个性化推荐系统综述[J]．计算机工程与应用，2012, 48(07):66-76.

[2] Bawden D, Robinson L. The dark side of information: overload, anxiety and other paradoxes and pathologies[J]. Journal of Information Science, 2009, 35(2):180-191.

[3] 孙鲁平，张丽君，汪平．网上个性化推荐研究述评与展望[J]．外国经济与管理, 2016, 38(06):82-99.

[4] 刘建国，周涛，汪秉宏．个性化推荐系统的研究进展[J]．自然科学进展，2009, 19(01):1-15.

[5] Sharma A, Hofman J M , Watts D J. Estimating the Causal Impact of Recommendation Systems from Observational Data[C] 16th ACM Conference on Economics and Computation. ACM, 2015:453-470.

[6] Linden G, Smith B, York J. Amazon.com Recommendations: Item-to-Item Collaborative Filtering[J]. IEEE Internet Computing, 2003, 7(1):76-80.

[7] Davidson J, Liebald B, Liu J, et al. The YouTube video recommendation system[C] ACM Conference on Recommender Systems. ACM, 2010:293-296.

[8] Gomez-Uribe C A, Hunt N. The Netflix Recommender System: Algorithms, Business Value, and Innovation[J]. Acm Transactions on Management Information Systems, 2015, 6(4):1-19.

[9] 卜旭松．基于物品协同过滤的个性化视频推荐算法改进研究[D]．宁夏大学, 2015.

[10] 李姗姗．基于协同过滤的视频推荐系统设计[D]．南京邮电大学, 2017.

[11] 熊玲．基于标签的教学视频推荐系统的设计与实现[D]．华中师范大学，2015.

[12] 范波，程久军．用户间多相似度协同过滤推荐算法[J]．计算机科学，2012, 39(01):23-26.

[13] 狄博，王晓丹．基于 Python 语言的面向对象程序设计课程教学[J]．计算机工程与科学，2014，36(S1):122-125.

[14] 龚新定，余艳梅，吴小强等．基于Django的实验室信息管理系统设计[J]. 微型机与应用，2016，35(22):108-111.

[15] 张玉洁，杜雨露，孟祥武．组推荐系统及其应用研究[J]．计算机学报，2016, 39(04):745-764.

[16] Pazzani M J, Billsus D. Content-Based Recommendation Systems[M] The adaptive web. Springer-Verlag, 2007:325-341.

[17] 许海玲，吴潇，李晓东等．互联网推荐系统比较研究[J]．软件学报，2009, 20(02):350-362.

[18] Sarwar B, Karypis G, Konstan J, et al. Item-based collaborative filtering recommendation algorithms[C] International Conference on World Wide Web. ACM, 2001:285-295.

[19] 武永亮，赵书良，李长镜等.基于 TF-IDF 和余弦相似度的文本分类方法[J].中文信息学报，2017, 31(05):138-145.

[20] Lin W, Alvarez S A, Ruiz C. Efficient Adaptive-Support Association Rule Mining for Recommender Systems[J]. Data Mining & Knowledge Discovery, 2002, 6(1):83-105.

[21] Burke, Robin. Hybrid Recommender Systems: Survey and Experiments[J]. User Modeling and User-Adapted Interaction, 2002, 12(4):331-370.

[22] 熊斌．基于 B/S 三层架构 OA 系统的设计和实现[D]．电子科技大学，2011.

[23] 孔维梁．协同过滤推荐系统关键问题研究[D]．华中师范大学，2013.

[24] 孙红亮．基于三层架构的校园网站设计与实现[D]．河北师范大学，2014.

致　　谢

四年的大学时间很快，我们马上毕业了，就连最后的毕业论文也写到结尾了。在这里，我向在这四年里帮助过我，激励过我的老师，同学们表达我最诚挚的感谢。

在推荐系统方面，以前了解的几乎为零，但是希望自己可以多学习，多了解，因此选择了个性化视频推荐系统，首先非常感谢我的导师肖迎元教授愿意让我做此课题。从论文的开题到最终的定稿，肖老师一直在给我们精心的指导。选题之后就开始指导我们怎么查阅资料，阅读文献，为开题报告提前做好准备。在中期设计开发过程中，给我们提供思路以及算法的讲解。后期书写论文时，指导我们写作的规范，论文的书写结构等等。其次还有研究生的学长，给我们指导怎么获取数据。

在大四这一年，一直在北京做实习工作，学校的很多事情不能自己处理，基本都是我的舍友帮忙，还有在论文格式方面，也给与了很大帮助，在这里我非常感谢你们的帮助，让我能顺利完成论文和实习。

最后，我还要感谢我的爸爸妈妈和姐姐，在这十几年的求学生涯当中，是你们一直在背后默默支持我、鼓励我，我才能这么顺利的一路走来，谢谢你们。

祝大家身体健康，工作顺利。